LOGZILLA DOCUMENTATION

Zeek

LogZilla App Store application: Zeek

LogZilla App Store · Generated April 27, 2026 · logzilla.ai/docs/logzilla-appstore/zeek

Overview

Zeek (formerly Bro) is an open-source network security monitor that provides deep packet inspection and protocol analysis. Zeek generates structured logs for connections, DNS queries, HTTP requests, SSL/TLS handshakes, and security events, making it valuable for threat hunting and incident response.

App Function

- Parse Zeek JSON-format logs from the HTTP Event Receiver
- Extract network metadata (IPs, ports, domains, protocols)
- Categorize events by type (network, auth, security, system)
- Provide security-focused dashboard for threat hunting
- Alert on security notices, threat intel matches, and authentication failures

Vendor Documentation

- Zeek Log Files Reference (https://docs.zeek.org/en/master/script-reference/log-files.html)
- Zeek JSON Format Logs (https://docs.zeek.org/en/master/log-formats.html#zeek-json-format-logs)
- Zeek Logs Overview (https://docs.zeek.org/en/master/logs/index.html)

Device Configuration

Configure Zeek to output JSON-format logs:

Edit `/opt/zeek/share/zeek/site/local.zeek` and add:

```
@load policy/tuning/json-logs.zeek
```

Deploy the configuration:

```
zeekctl deploy
```

Verify JSON format is enabled:

```
tail -1 /opt/zeek/logs/current/conn.log
```

Verification

Confirm output is JSON format with fields like `id.orig_h`, `id.resp_h`, etc.

Log Ingestion

Zeek logs must be forwarded to LogZilla via the HTTP Event Receiver. See the HTTP Event Receiver documentation (https://www.logzilla.ai/docs/receiving-data/http-event-receiver) for setup instructions.

When forwarding Zeek logs, include these required extra fields:

Field	Value	Purpose
`_source_type`	`zeek`	Identifies logs for the Zeek app
`_source`	Log filename (e.g., `conn.log`)	Sets the `Zeek Area` tag

Configure syslog-ng to tail Zeek log files and forward to LogZilla. See the Syslog Relays documentation (https://www.logzilla.ai/docs/administration/syslog-relays) for detailed configuration examples.

Example syslog-ng source for Zeek logs:

```
source s_zeek_conn {
    file("/opt/zeek/logs/current/conn.log" flags(no-parse));
};
```

Example destination with required extra fields:

```
destination d_logzilla_zeek {
    http(
        url("https://<LOGZILLA_HOST>/incoming")
        method("POST")
        user-agent("syslog-ng Zeek Relay")
        headers(
            "Content-Type: application/json",
            "Authorization: token <YOUR_TOKEN>"
        )
        body-prefix("{\"events\": [\n")
        delimiter(",\n")
        body('$(format-json
            --pair message="$MESSAGE"
            --pair extra_fields.\_source_type="zeek"
            --pair extra_fields.\_source="conn.log"
        )')
```

```
        body-suffix("\n]}")
        batch-lines(1000)
        batch-bytes(1048576)
        batch-timeout(500)  # milliseconds - flush after 500ms even if batch not full
    );
};

log {
    source(s_zeek_conn);
    destination(d_logzilla_zeek);
    flags(flow-control);
};
```

Notes:

- The `batch-timeout(500)` setting ensures events are sent within 500ms even if the batch is not full. For low-volume environments, reduce `batch-lines` and `batch-bytes` or decrease `batch-timeout` to see events sooner.
- The `_source` field determines the `Zeek Area` tag and `Event Class`. Create separate source/destination pairs for each Zeek log type (conn.log, dns.log, http.log, etc.) or use syslog-ng's `wildcard-file()` source with the filename in the body.

Incoming Log Format

Zeek logs are single-line JSON objects. Example (conn.log):

```
{"ts":1641949013.67,"id.orig_h":"192.168.10.107","id.resp_h":"192.168.10.255",
"id.resp_p":53,"proto":"udp","service":"dns"}
```

Parsed Metadata Fields

Tag Name	Example	Description
`Vendor`	`Zeek`	Vendor identifier
`Product`	`NSM`	Product identifier (Network Security Monitor)
`Event Class`	`security`	Event classification (network, auth, security, system)
`Zeek Area`	`conn`	Log type (conn, dns, http, ssl, ssh, notice, etc.)
`SrcIP`	`192.168.1.100`	Source IP address

Tag Name	Example	Description
DstIP	10.1.1.50	Destination IP address
DstPort	https	Destination port (translated to service name)
Domain	example.com	Domain name from DNS or DHCP
Zeek RCode	NXDOMAIN	DNS response code
Zeek Status	OK	HTTP/SIP status message
Zeek Operation	NetrLogonSamLogon	DCE/RPC operation name
Zeek Auth Success	false	SSH authentication result
Zeek SSL Validation	self signed certificate	SSL certificate validation status
Zeek Notice	Scan::Port_Scan	Zeek notice type from notice.log
MitreId	T1046	MITRE ATT&CK technique ID
MITRE Tactic	Discovery	MITRE ATT&CK tactic name

MITRE ATT&CK Mappings

Zeek notice types are mapped to MITRE ATT&CK techniques:

Notice Type	Technique	Tactic
Scan::Port_Scan	T1046	Discovery
Scan::Address_Scan	T1046	Discovery
SSH::Password_Guessing	T1110	Credential Access
FTP::Bruteforcing	T1110	Credential Access
SSL::Invalid_Server_Cert	T1557	Credential Access

Notice Type	Technique	Tactic
`HTTP::SQL_Injection_Attacker`	T1190	Initial Access
`TeamCyru::Malware_Hash_Registry`	T1204	Execution
`DNS::Tunneling`	T1071	Command and Control
`Conn::Large_Outbound_Transfer`	T1048	Exfiltration

Log Examples

Connection Log


```
{"ts":1591367999.305988,"uid":"CMdzit1AMNsmfAIiQc","id.orig_h":"192.168.4.76",
"id.orig_p":36844,"id.resp_h":"192.168.4.1","id.resp_p":53,"proto":"udp",
"service":"dns","conn_state":"SF"}
```


DNS Query


```
{"ts":1591368000.123,"uid":"CX1l7X34hCbGkWGlB6","id.orig_h":"192.168.4.76",
"id.resp_h":"192.168.4.1","query":"example.com","rcode_name":"NOERROR"}
```


SSH Authentication Failure


```
{"ts":1591368100.456,"uid":"CzEmsljW9ooL0WnBd","id.orig_h":"10.0.0.50",
"id.resp_h":"192.168.4.37","id.resp_p":22,"auth_success":false,
"auth_attempts":3,"client":"SSH-2.0-OpenSSH_7.9p1"}
```


Security Notice


```
{"ts":1591368200.789,"note":"Scan::Port_Scan","msg":"192.168.1.100 scanned at
least 15 unique ports of host 10.0.0.1","src":"192.168.1.100"}
```

Dashboards

Dashboard	Description
Zeek: Security	Security KPIs, MITRE analysis, notices, auth failures
Zeek: Network	Traffic analysis, DNS, connections, services

Triggers

Trigger	Description
Zeek: MITRE ATT&CK Threat Detected	Events with MITRE technique mapping
Zeek: Security Notice Detected	Zeek built-in detection alerts
Zeek: Threat Intel Match	Threat intelligence indicator matches
Zeek: Signature Match	IDS-style signature detections
Zeek: Protocol Anomaly Detected	Unusual network behavior (weird logs)
Zeek: SSH Authentication Failure	Failed SSH authentication attempts
Zeek: SSL Certificate Problem	Certificate validation failures (no notify)
Zeek: DNS NXDOMAIN Response	Failed DNS lookups (no notify)